MÉTODO DE INGENIERÍA

Danna A. Espinosa

Carlos J. Bolaños

Cristian F. Perafan

Computación y Estructuras Discretas I

Universidad Icesi

Profesor: Uram Anibal Sosa

ÍNDICE

Contexto del problema:

Debido a la gran admiración que sienten los estudiantes del curso de **Computación y Estructuras Discretas I** hacía los juegos y a la serie **Phineas y Ferb**, se les ha encargado desarrollar el siguiente juego:

Phineas y Ferb construyen una máquina de teletransportación sobre el planeta tierra e invitan a sus amigos y conocidos a probarla (Isabella, Baljeet, Buford, Candace, Jeremy, Stacy y Vanessa). Debido a que en el mundo hay tantos países y que Phineas y Ferb no querían terminar en medio de una guerra, estos han decidido limitar el número de ciudades de destino, teniendo en lista solo 50, los cuales son:

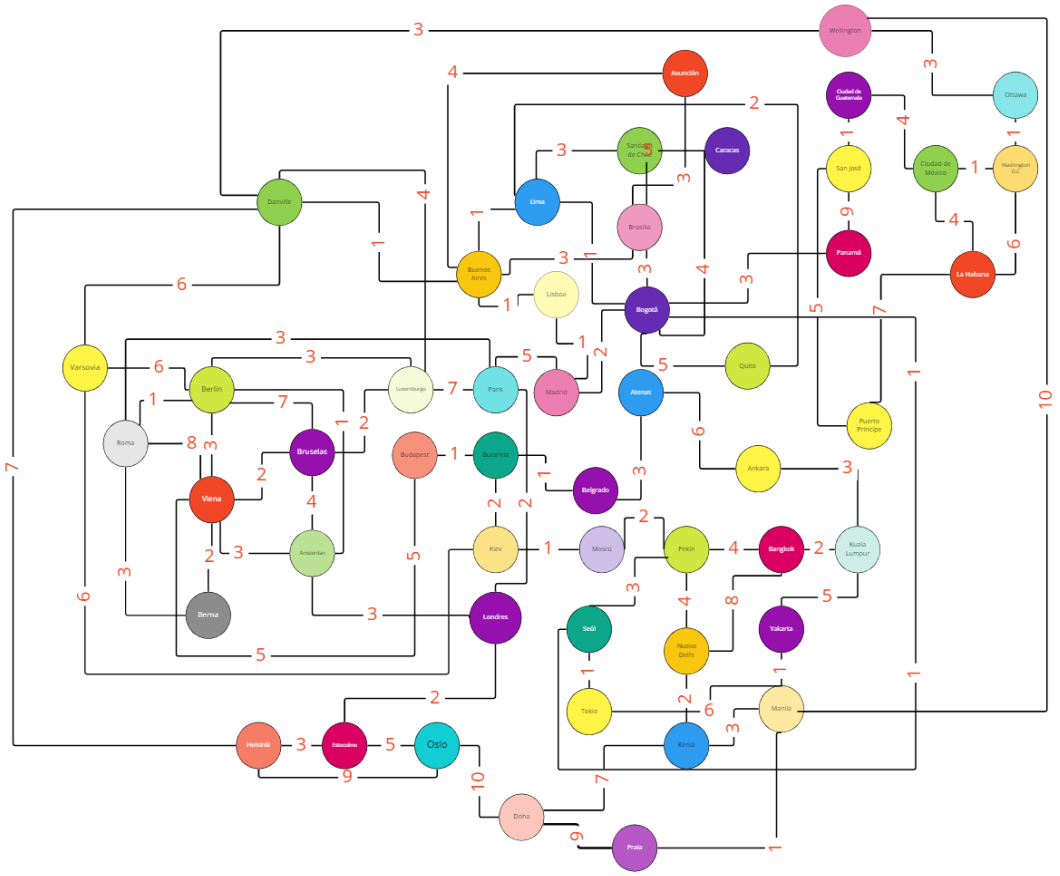
Lista de ciudades:
Danville, Área Limítrofe
Buenos aires, Argentina
Berlín, Alemania
Bruselas, Bélgica
Brasília, Brasil
La Habana, Cuba
Quito, Ecuador
Bogotá, Colombia
Moscú, Rusia
Varsovia, Polonia
Lisboa, Portugal
Londres, Reino Unido
Bucarest,Rumanía
Belgrado, Serbia
Manila,Filipinas
París, Francia
Bangkok, Tailandia
Ankara, Turquía
Pekín, China
Santiago de Chile, Chile
Panamá, Panamá
Lima, Perú
Asunción, Paraguay
Tokio, Japón
Ámsterdam, Países Bajos
Doha, Catar
Praia, Cabo Verde
Ottawa, Canadá
Viena, Austria

Kuala Lumpur, Malasia
Ciudad de México, México
Roma, Italia
Washington D.C, Estados Unidos
Helsinki, Finlandia
Estocolmo, Suecia
Berna, Suiza
Ciudad de Guatemala, Guatemala
Atenas, Grecia
Puerto Príncipe, Haití
Kiev, Ucrania
Nueva Delhi, India
Yakarta, Indonesia
Nairobi, Kenia
San José, Costa Rica
Seúl, Corea Del Sur
Oslo, Noruega
Madrid, España
Budapest, Hungría
Wellington, Nueva Zelanda
Caracas, Venezuela

Al momento de probarla, van ingresando a la máquina uno a uno, pero lo que no esperaban es que la máquina fallaría y dejaría a cada uno de sus amigos en distintas ciudades del mundo. Por suerte, Phineas y Ferb se dan cuenta antes de ingresar a la máquina debido a un dispositivo que muestra la ubicación de cada uno de estos, por lo que el nuevo reto de Phineas y Ferb es encontrar a cada uno de sus amigos de una manera eficiente, es decir, en el camino más corto en términos de tiempo. Para ello, construyen una máquina nueva (modo pistola) que les permitirá viajar por las diferentes ciudades. Se debe tener en cuenta que en algunos casos para llegar a una ciudad en específico Phineas y Ferb deberán atravesar otras ciudades, además, cuando se atraviesa cada portal se deberán tardar entre 1 minuto hasta 10 minutos, por lo cual ellos no saben que ruta seguir para llegar más rápido a sus amigos, para lograrlo deberán mejorar su dispositivo de tal manera que muestre la ruta más corta en términos de tiempo y poder salvar a sus amigos.

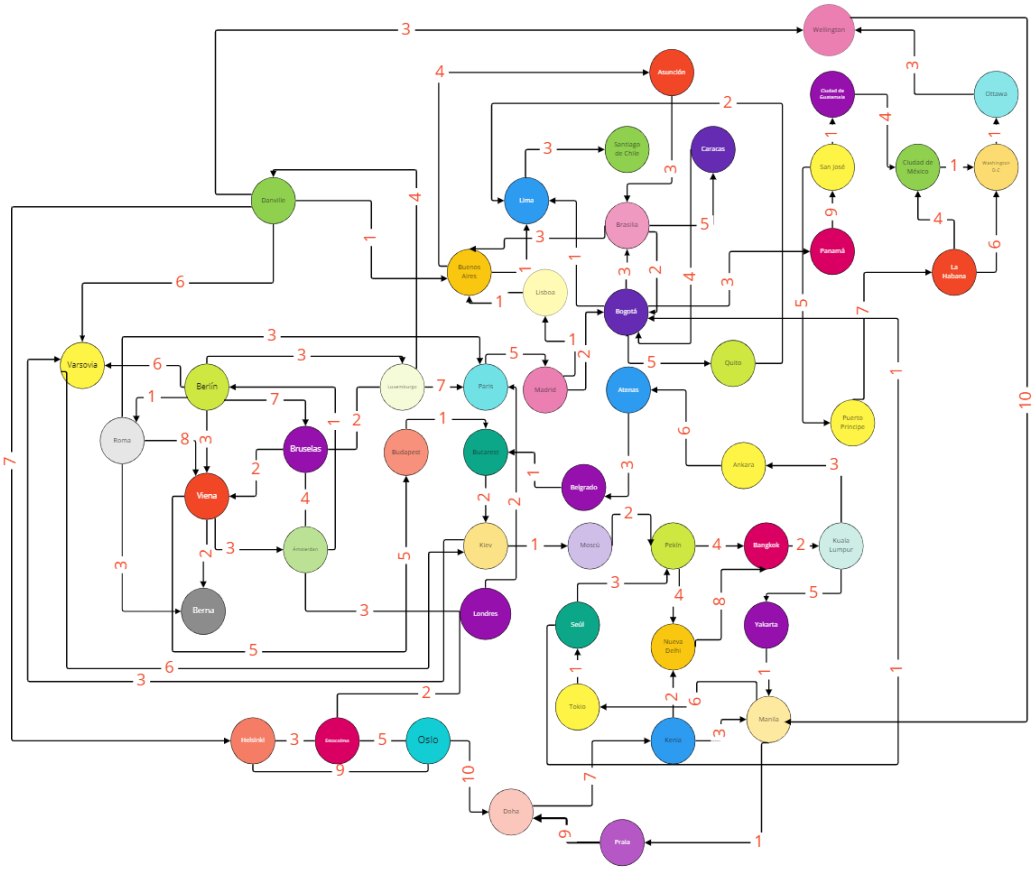
Grafos del problema:

Grafo 1-No Dirigido:



Img 1. Los **vértices** son las **<u>ciudades</u>** en las que pueden teletransportarse y las **aristas** son las **<u>conexiones que tienen entre las ciudades</u>**, habiendo una arista de conexión entre las ciudades cercanas y levemente cercanas, y así mismo depende los **pesos**, donde los pesos serán **<u>el número de minutos que se demoren en llegar a la otra ciudad.</u>**

Grafo 2-Dirigido:



Img 2. Los **vértices** son las **ciudades** en las que pueden teletransportarse y las **aristas** son las **conexiones que tienen entre las ciudades**, habiendo una arista de conexión entre las ciudades cercanas y levemente cercanas, y así mismo depende los **pesos**, donde los pesos serán **el número de minutos que se demoren en llegar a la otra ciudad.** Las ciudades donde se podrán dirigir, estarán definidas con una arista dirigida.

Identificación del problema:

Se requiere trabajar en la mejora del dispositivo GPS que tienen Phineas y Ferb de tal manera que muestre el camino más corto en términos de tiempo para salvar a sus amigos.

Para ello deben usar la teoría de grafos e implementar los grafos 1 y 2 antes descritos usando al menos dos algoritmos de grafos.

Se debe implementar los algoritmos Dijkstra y Floyd-Warshall

Recopilación de la información:

Grafo:

Un grafo en el ámbito de las ciencias de la computación es un tipo abstracto de datos (TAD), que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos. Los grafos tienen diferentes tipos de implementaciones: con una matriz de adyacencias (forma acotada) y con listas y multilistas de adyacencia (no acotadas).

Fuentes:
Explained: Graphs - MIT News
Grafo (tipo de dato abstracto)

Problema del camino mínimo:

En teoría de grafos, el problema del camino mínimo es el problema de encontrar un camino entre un par de vértices de un grafo, de tal manera que la suma de los pesos de las aristas que recorre el camino sea lo menor posible.

Los algoritmos más importantes para solucionar este problema son:

- Algoritmo de Dijkstra
- Algoritmo de Floyd-Warshall

Fuentes:
Shortest path problem

Algoritmo de Floyd-Warshall:

El algoritmo de Floyd-Warshall es un algoritmo para encontrar el camino mínimo entre todos los pares de vértices en un grafo ponderado, el algoritmo no funciona para grafos con pesos negativos ya que caería en un ciclo infinito.

Pseudocódigo:

```
n = no of vertices
A = matrix of dimension n*n
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            A^k[i, j] = min (A^k-1[i, j], A^k-1[i, k] + A^k-1[k, j])
return A
```

Fuentes:
Floyd-Warshall Algorithm

Algoritmo de Dijkstra´s:

El algoritmo de Dijkstra´s nos permite encontrar la distancia mínima entre cualquier par de vértices de un grafo, el algoritmo de Dijkstra´s se utiliza para:

- Para encontrar el camino mínimo.
- En aplicaciones de redes sociales.
- En redes de telefonía.
- Para encontrar ubicaciones en un mapa.

Pseudocódigo:

```
function dijkstra(G, S)
    for each vertex V in G
        distance[V] <- infinite
        previous[V] <- NULL
        If V != S, add V to Priority Queue Q
    distance[S] <- 0

    while Q IS NOT EMPTY
        U <- Extract MIN from Q
        for each unvisited neighbour V of U
            tempDistance <- distance[U] + edge_weight(U, V)
            if tempDistance < distance[V]
                distance[V] <- tempDistance
                previous[V] <- U
    return distance[], previous[]
```

Fuentes:
Dijkstra's Algorithm

Búsqueda de ideas creativas:

- Alternativa 1: Para cubrir las necesidades que ha generado el problema descrito anteriormente , una de las posibles soluciones consistiría en desarrollar una aplicación en java que funcione por consola y simule el desarrollo del juego. Como parte de esta aplicación en java también se agregarían todas las estructuras de datos necesarios para el cumplimiento de los requerimientos del juego (Estructura genérica para grafos).
- Alternativa 2: Para el desarrollo del juego, existe la posibilidad de desarrollar una interfaz gráfica en javafx que permita modelar y mostrar la ubicación de cada uno de los amigos de Phineas y Ferb. Se debe combinar con la implementación de grafo genérica.
- Alternativa 3: La implementación será construida en javafx, contendrá elementos que permitan el desplazamiento por medio de los portales. La modelación de la ubicación de los amigos de Phineas y Ferba, y la de las distancias entre cada ciudad capital será almacenada en un árbol binario, lo cual permitirá acceder a la información necesaria para calcular los caminos mínimos de una manera eficiente.
- Alternativa 4: Para cubrir las necesidades que se han generado a partir de la problemática, una posible solución sería desarrollar una aplicación en java que funcione por consola, donde la estructura base para el funcionamiento del juego sea de listas doblemente enlazadas, en donde se haga la búsqueda de cada uno de los amigos de Phineas y Ferb y se agreguen a una estructura de datos como cola o pilas a medida que se encuentren.
- Alternativa 5: Para cubrir las necesidades que ha generado el problema descrito anteriormente, una de las posibles soluciones consistiría en desarrollar una aplicación en java que funcione mediante una interfaz gráfica, en donde el usuario escoja el punto de partida (Donde se encuentran Phineas y Ferb) y el punto de finalización (La ciudad en donde se encuentre alguno de sus amigos), finalmente se emplearán los dos tipos de grafos con algoritmos como Dijkstra y Floyd Warshall y se mostrará en pantalla la ruta más corta en términos del tiempo.
- Alternativa 6: La solución será implementada con JavaFx para que permita una buena interacción con el usuario. Hablando desde la parte de la lógica del código, los países serán almacenados en un HashMap, donde el key corresponde al nombre del país, el value será un objeto país, a su vez, este último contendrá a los personajes del juego en caso de que se ubique de manera aleatoria ahí.

Transición de la formulación

Alternativas descartadas:

- Alternativa 3: Esta idea es descartada, porque a pesar de que la solución permite un acceso rápido a la información necesaria de cada una de las distancias entre los países,

la validación de la aristas que existen entre cada uno de los vértices así como su peso, representan una gran dificultad a la hora del desarrollo del juego.

- Alternativa 4: Esta idea es descartada, debido a que la estructura de datos que es utilizada en esta solución no es la más apropiada a la hora de representar los diferentes escenarios que se plantean en la problemática.
- Alternativa 6: Esta idea es descartada, debido a que la estructura de datos que es utilizada en esta solución no es la más apropiada para poder dar solución a las necesidades creadas por la problemática, además esta solución no sería reutilizable para futuros proyectos, esto debido a que utiliza estructuras que no son propias de la solución.

Diseño preliminar:

Rúbrica para evaluar ideas:

- ***Soluciona el problema:*** *La idea implementación soluciona el problema totalmente.*
- ***Usabilidad:*** *La idea implementada es de fácil usabilidad, es decir, es entendible para el usuario*
- ***Mantenibilidad:*** *La idea implementada es de fácil mantenibilidad*
- ***Reutilización:*** *La idea implementada permite la fácil reutilización del código para otros posibles contextos del problema*
- ***Uso de estructuras adecuadas:*** *La idea implementada usa las mejores estructuras adecuadas para el desarrollo de la solución:*

Evaluación	Soluciona el problema	Usabilidad	Mantenibilidad	Reutilización	Uso de estructuras adecuadas
Alternativa					

Calificación: de 1 a 5, siendo uno la calificación más baja y 5 la más alta

Evaluación de la mejor solución:

Evaluación	Soluciona el problema	Usabilidad	Mantenibilidad	Reutilización	Uso de estructuras adecuadas	Total
Alternativa 1	2	2	2	4	5	15
Alternativa 2	1	4	3	2	5	15
Alternativa 3	1	4	3	3	1	12
Alternativa 4	1	2	2	3	1	9
Alternativa 5	5	5	4	4	5	**23**
Alternativa 6	1	4	3	1	1	10

Según la evaluación de cada una de las alternativas, con un total de 23 puntos, la mejor solución es la **alternativa 5**, puesto que esta tiene una mejor aproximación a lo que se busca resolver con el programa y se hace el uso de la estructura correcta (en este caso es el grafo).